Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

Prozedurale Generierung von Baumstrukturen innerhalb der Unreal Engine 4

Bachelor Abschlussarbeit

David Liebemann, 17.03.17

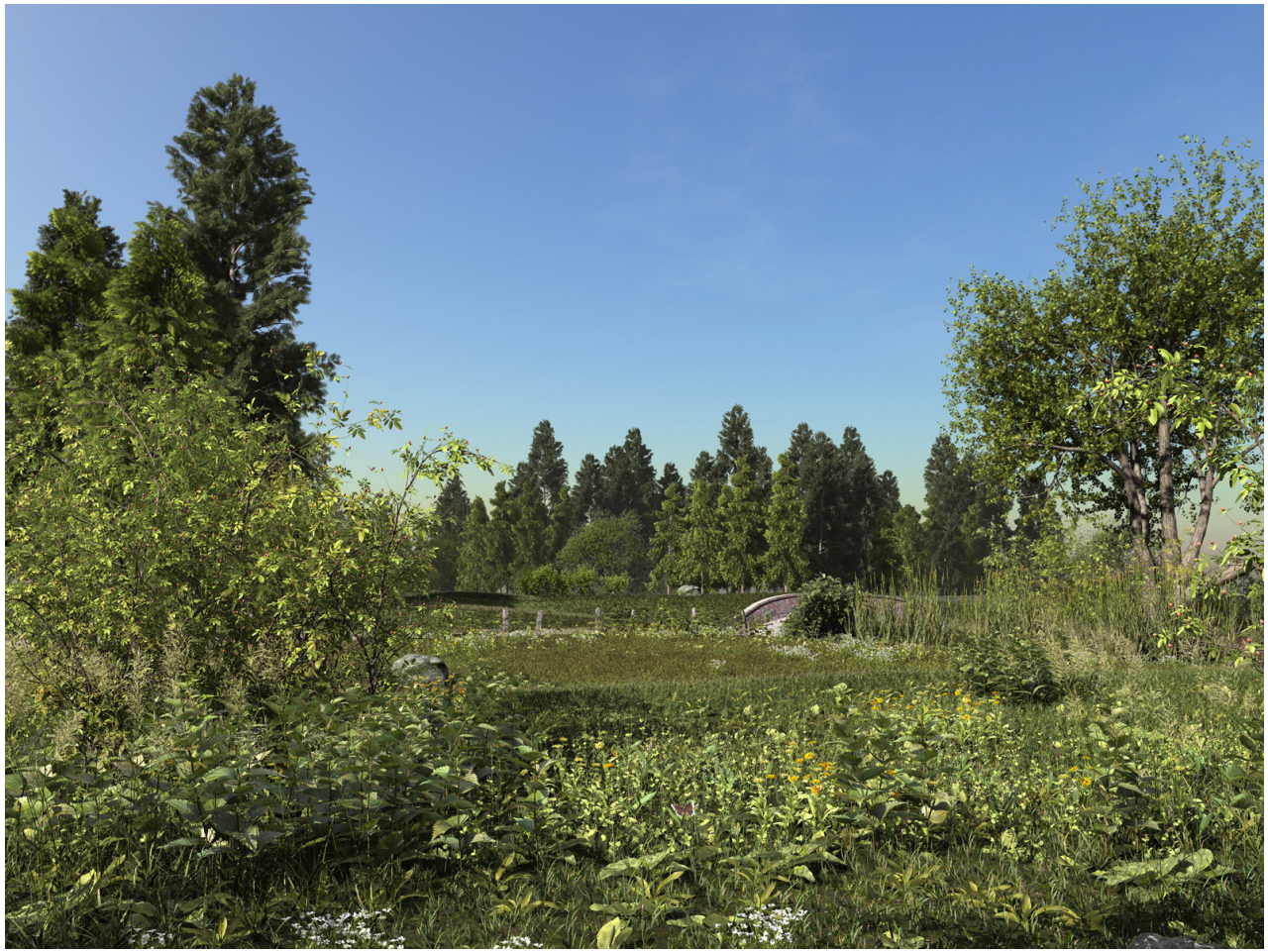
Überblick

1. Einleitung
2. Lindenmayer-Systeme
3. Space Colonization Algorithmus
4. Implementierung
5. Ergebnisse
6. Zusammenfassung und Ausblick
7. Quellen

1 Einleitung

Prozedurale Generierung

- Konstruktion von 3D-Modellen durch computergenerierte Daten
- Benötigt eingeschränkten Eingriff durch Benutzer
- Generierung von Pflanzenmodellen ist ein wichtiger Bestandteil
- In dieser Arbeit: Konzentration auf die Generierung von Baumstrukturen



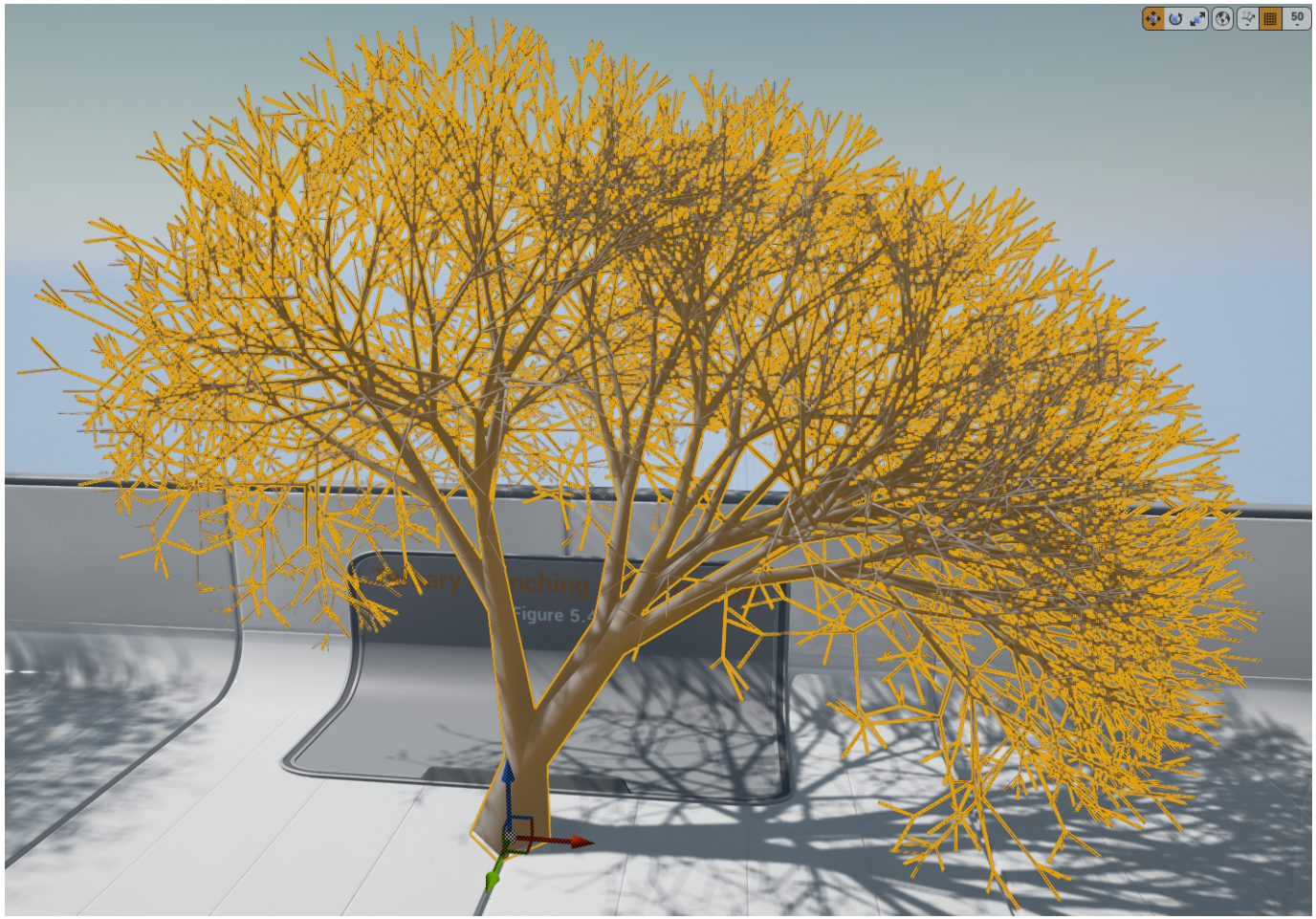
Prozedural generierte Landschaftsszene. [Gre]

Ansatz

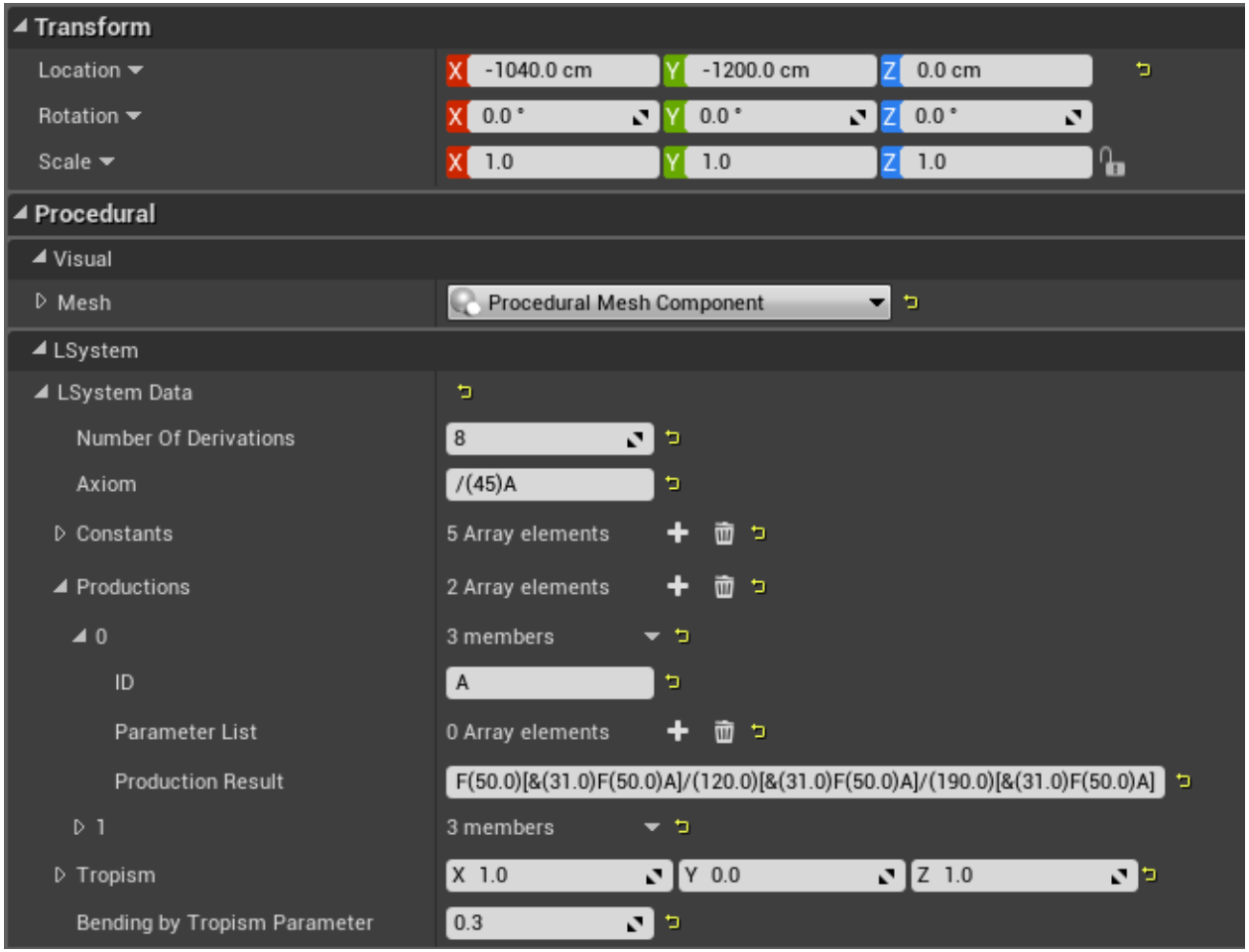
- Implementierung von zwei Verfahren zur prozeduralen Generierung von Baumstrukturen:
  - Lindenmayer-Systeme
  - Space Colonization Algorithmus
- Verwendung des Frameworks „Unreal Engine 4“
- Vereinfachte Darstellung von Ästen in Form von Zylindern

Unreal Engine 4

- Sammlung von Softwarewerkzeugen
- In C++ programmiert mit frei zugänglichem Quellcode
- Inhalte werden in C++ oder Blueprint erstellt und leiten von Framework-Basisklassen ab
- Verfügbarkeit eines visuellen Editor:
  - Ermöglicht die einfache Positionierung von Actors
  - Erlaubt die Eingabe von Parametern über das Editor-UI



Positionierung eines Actors im Leveleditor



Eingabefenster für Actor-Parameter

2 Lindenmayer-Systeme

- Von Aristid Lindenmayer 1968 entwickelte Erweiterung von Ersetzungssystemen
- Weitere Ergänzungen durch Prusinkiewicz und Lindenmayer in 1990
- Funktionsweise basiert auf der Ersetzung von Zeichen in Zeichenketten
- Grafische Interpretation der Resultate ergibt Modelldaten

2.1 D0L-Systeme

Definition D0L-System: 2.1.1 *Ein deterministisches, kontextfreies L-System (D0L-System) ist ein Tupel* $G = (V, P, \omega)$ *mit:*

$\boldsymbol{V}$ **:** *Ein nichtleeres, endliches Alphabet*

$\boldsymbol{P}$ **:** *Eine Menge von Produktionsregeln in der Form* $P : a \to b$ *mit* $a \in V$ *und* $b \in V^*$

$\boldsymbol{\omega \in V^+}$ **:** *Das Axiom, Startwort des L-Systems*

Verwendete Begriffe:

Deterministisch: Es existiert genau eine Produktionsregel für jedes der Symbole in V

Kontextfrei: Ersetzung findet unabhängig von umgebenden Symbolen statt

Ableitung: Gleichzeitige Ersetzung aller Symbole eines Wortes anhand der Produktionsregeln

Beispiel: Simulation des Wachstums der Blaualgen-Gattung „Anabaena“

$\boldsymbol{V}$ besteht aus den Symbolen $\{a_l, a_r, b_l, b_r\}$

a und b: Größe und Teilungsbereitschaft einer Zelle

l und r: Zellenpolarität

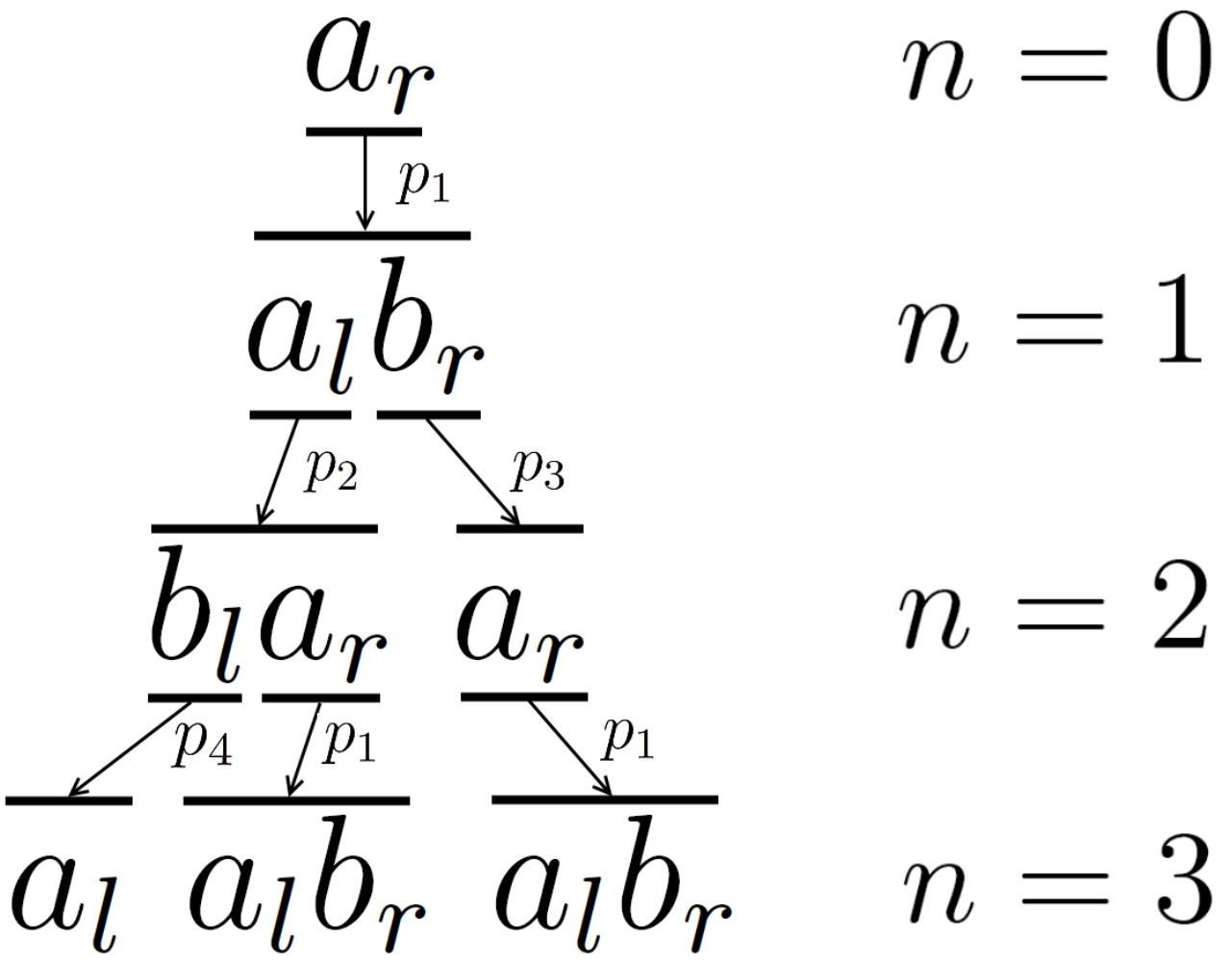
$\boldsymbol{P}$ besteht aus:

$p_1: \; a_r \rightarrow a_l b_r$

$p_2: \; a_l \rightarrow b_l a_r$

$p_3: \; b_r \rightarrow a_r$

$p_4: \; b_l \rightarrow a_l$


a_r
$n = 0$
p_1
$a_l b_r$
$n = 1$
p_2
p_3
$b_l a_r$ a_r
$n = 2$
p_4
p_1
p_1
a_l $a_l b_r$ $a_l b_r$
$n = 3$

2.2 Parametrische L-Systeme

- Erweiterung der D0L-Systeme

- Verwendung von parametrischen Wörtern anstatt einfacher Symbole:

  $\boldsymbol{A(a_1, ..., a_n)}$ **:** Parametrisches Wort mit $A \in V$ und $a_1, ..., a_n \in \Sigma$

  $\boldsymbol{\Sigma}$ **:** Menge formaler Parameter

- Im Nachfolger können arithmetische Ausdrücke anstelle formaler Parameter verwendet werden:

  $\boldsymbol{E(\Sigma)}$ **:** Arithmetischer Ausdruck

Parametrisches L-System: 2.2.1 *Ein Parametrisches L-System ist ein Tupel* $G = (V, \Sigma, P, \omega)$ *mit:*

$\boldsymbol{V}$ **:** *Ein nichtleeres, endliches Alphabet*

$\boldsymbol{\Sigma}$ **:** *Eine Menge formaler Parameter*

$\boldsymbol{P}$ **:** *Eine Menge von Produktionsregeln* $P : (V \times \Sigma^*) \rightarrow (V \times E(\Sigma)^*)^*$

$\boldsymbol{\omega} \in \boldsymbol{M}^+$ *mit* $M = (V \times \mathbb{R}^*)$ *– das Axiom in Form eines nichtleeren, parametrischen Wortes*

Beispiel: Definition und Ableitung eines parametrischen L-Systems

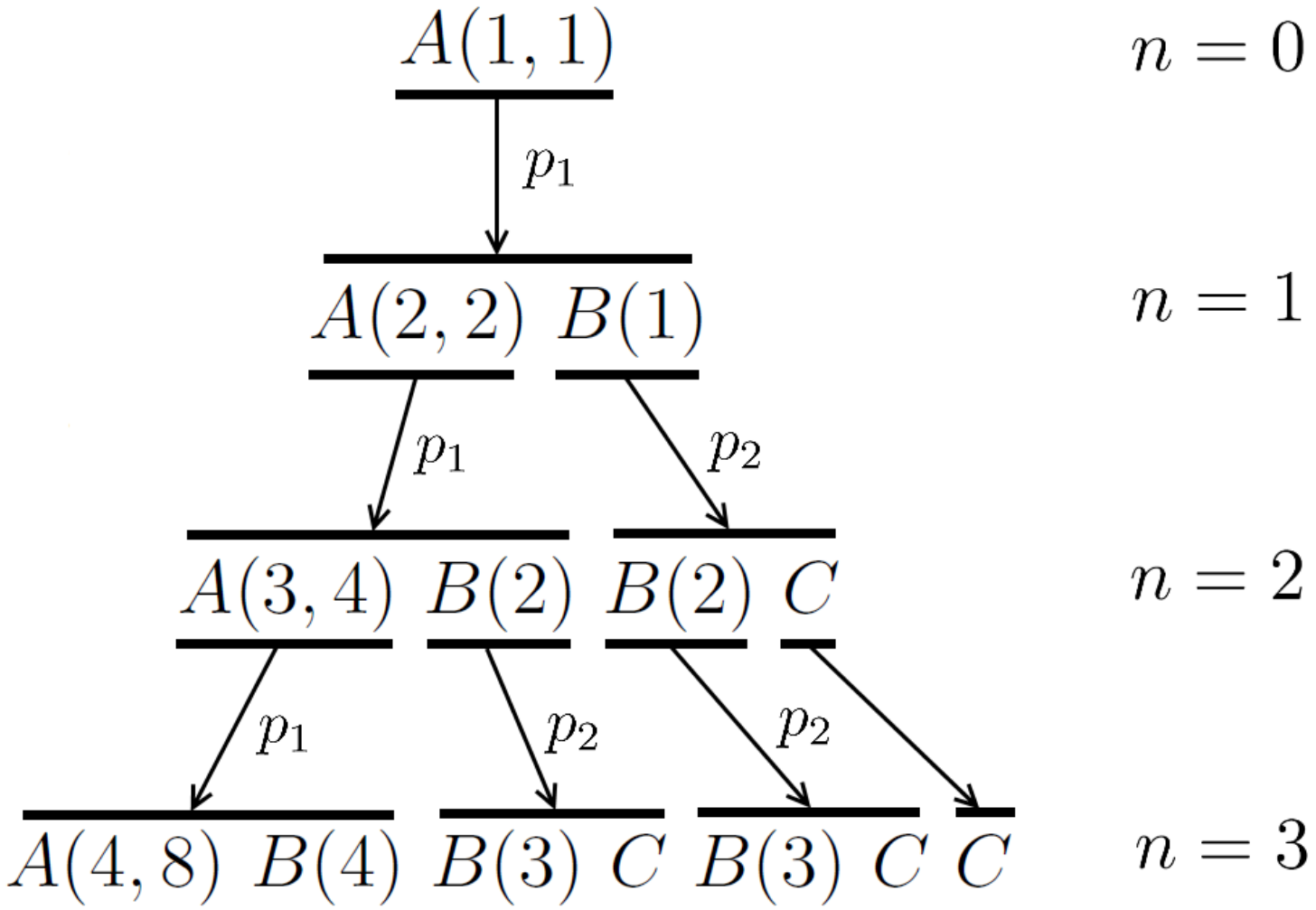
$$\begin{aligned} \omega &: A(1,1) \\ p_1 &: A(x,y) \rightarrow A(x+1, y*2)\ B(y) \\ p_2 &: B(x) \rightarrow B(x+1)\ C \end{aligned} \tag{1}$$

$\boldsymbol{V}$ besteht aus den Symbolen $\{A, B, C\}$

$\boldsymbol{\Sigma}$ besteht aus den formalen Parametern $\{x, y\}$

$\boldsymbol{P}$ besteht aus Produktionsregeln $\{p_1, p_2\}$

$\boldsymbol{\omega}$ entspricht $A(x,y)$ mit $x = 1$ und $y = 1$


A(1, 1)
n = 0
p1
A(2, 2) B(1)
n = 1
p1
p2
A(3, 4) B(2) B(2) C
n = 2
p1
p2
p2
A(4, 8) B(4) B(3) C B(3) C C
n = 3

2.3 Grafische Interpretation von L-Systemen

- Rückgabewerte von L-System-Ableitungen sind Zeichenketten, keine Modelldaten
- Grafische Interpretation von Zeichenketten: Turtle-Interpretation
- Zustand der Turtle ist ein Tupel $(\overrightarrow{p}, \overrightarrow{H})$ mit:

  $\overrightarrow{p}$: Position der Turtle

  $\overrightarrow{H}$: Blickrichtung (Heading) der Turtle

Turtle-Aktionen:

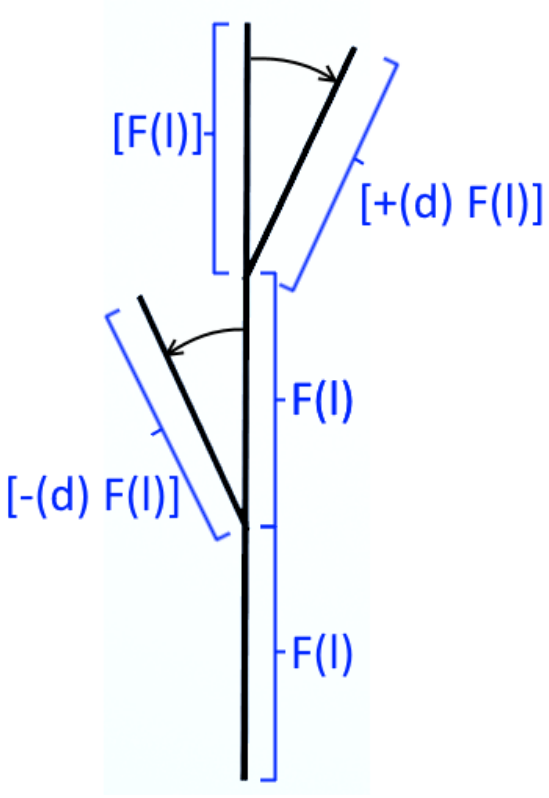
$\boldsymbol{F(l)}$ **:** Bewegung um $l > 0$ in Blickrichtung, Aktualisierung der Position und Zeichnung einer Linie

$\boldsymbol{+(d)}$ **:** Drehung um den Winkel d nach links, Aktualisierung der Blickrichtung

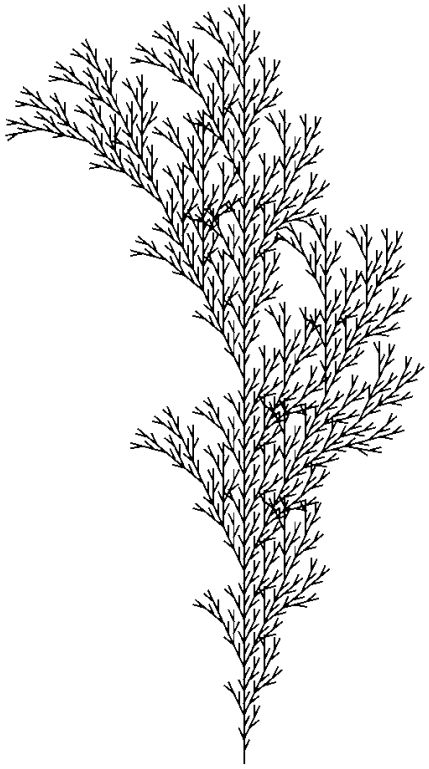
$\boldsymbol{-(d)}$ **:** Drehung um den Winkel d nach rechts, Aktualisierung der Blickrichtung

[**:** Ablage des Zustands auf einem Stack

] **:** Entnahme des obersten Zustands vom Stack und Aktualisierung des aktuellen Zustands



$n = 1,\ l = 240,\ d = 25°$



$n = 5,\ l = 15,\ d = 25°$

$$\omega \ : F(l)$$
$$p_1 \ : F(l) \rightarrow F(l)\ [-(d)\ F(l)]\ F(l)\ [+(d)\ F(l)]\ [F(l)]$$

2.4 Anpassungen für Baumstrukturen

Erweiterung der Turtle-Interpretation in den dreidimensionalen Raum

- Zustand der Turtle ist ein Tupel $(\overrightarrow{p}, \boldsymbol{R})$ mit:

  $\overrightarrow{\boldsymbol{p}}$: Position der Turtle

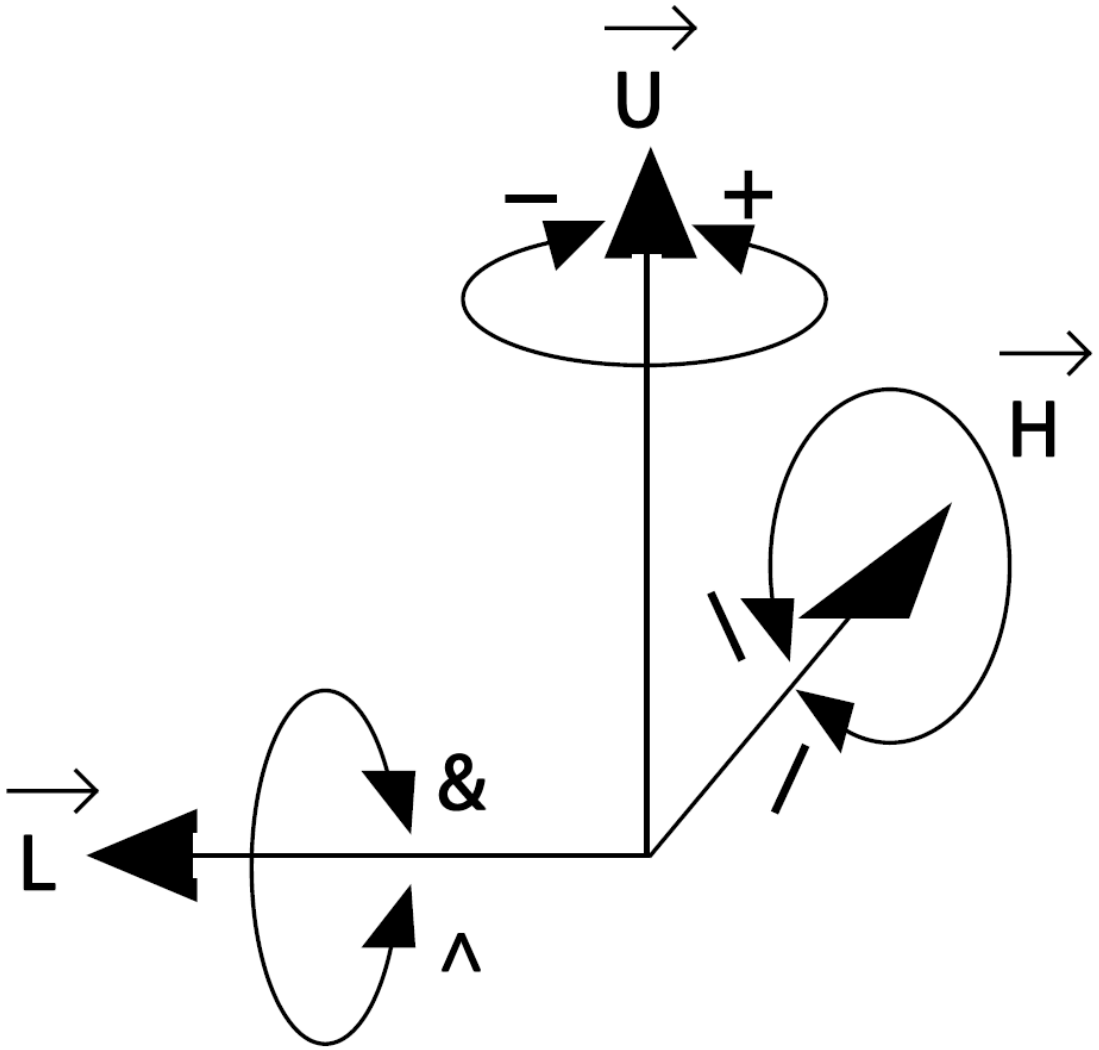
  $\boldsymbol{R}$: Rotationsmatrix der Turtle

- Einheitsvektoren $\overrightarrow{H}, \overrightarrow{L}, \overrightarrow{U}$ bilden das lokale Koordinatensystem der Turtle:

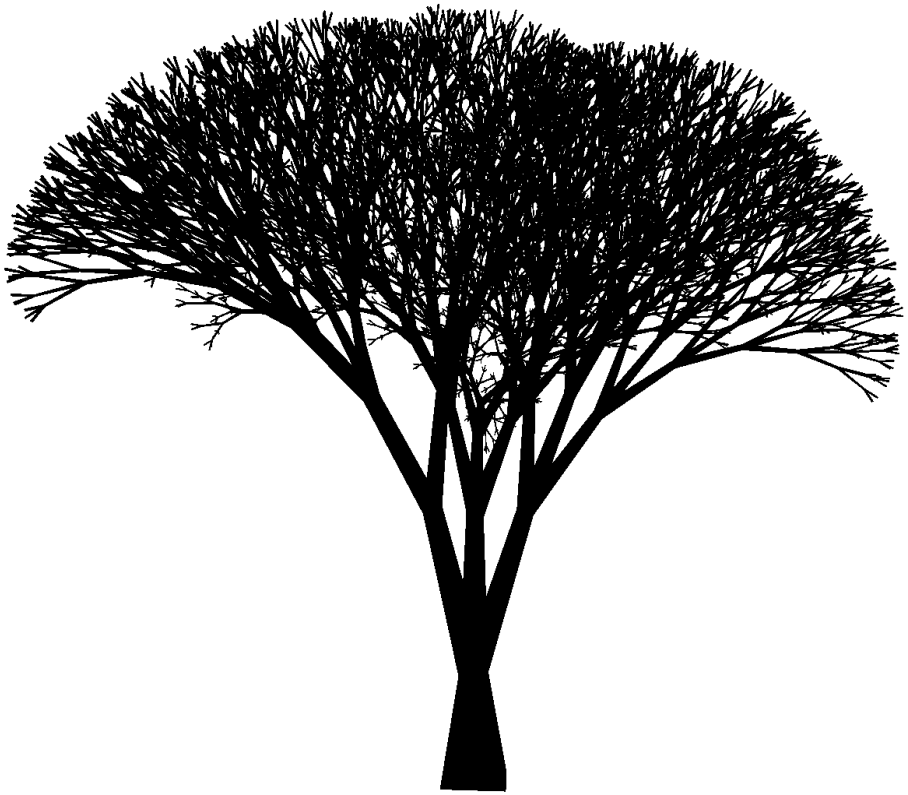
  $\overrightarrow{\boldsymbol{H}}$: Heading-Vektor

  $\overrightarrow{\boldsymbol{L}}$: Left-Vektor

  $\overrightarrow{\boldsymbol{U}}$: Up-Vektor



[PL90, S.19]

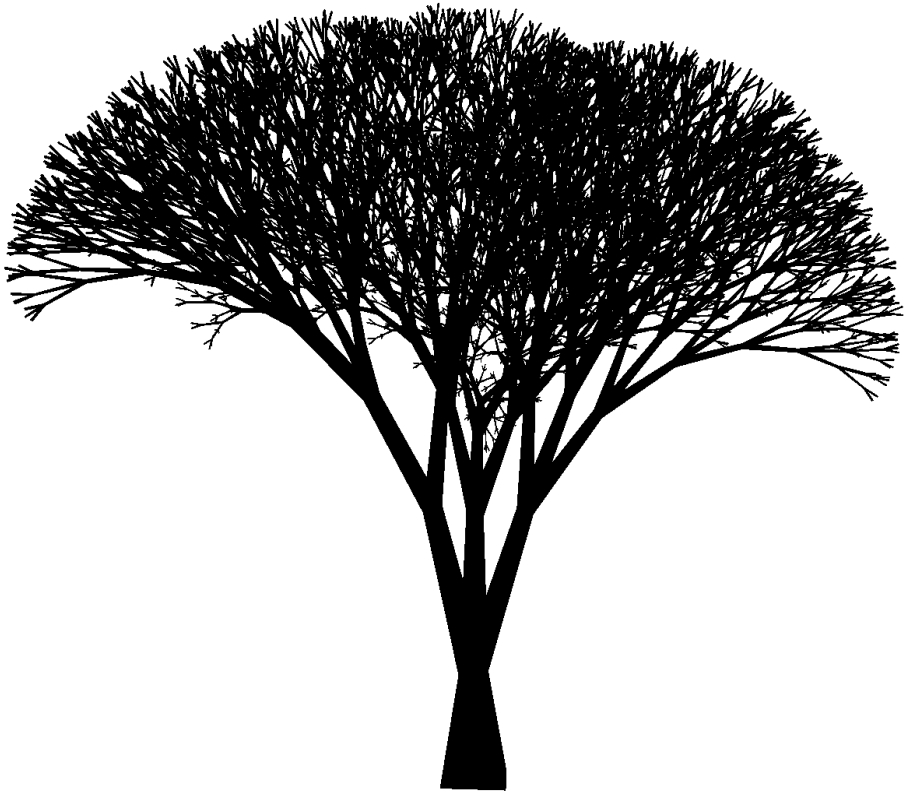


$n = 8,\ l = 50,\ d = 137.5°,$
$a = 18.95°,\ l_r = 1.3$

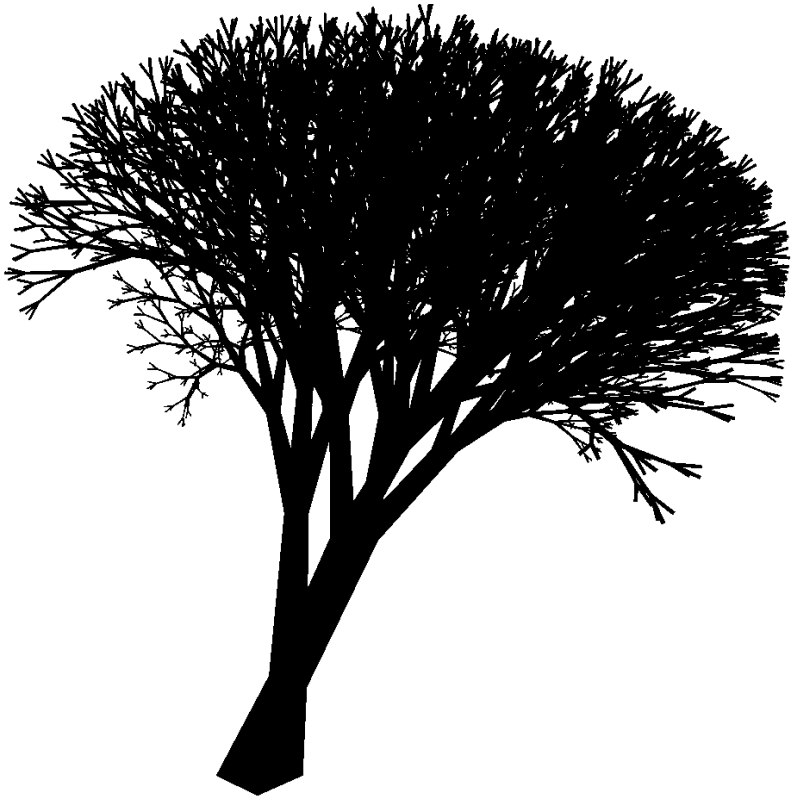
$$
\begin{aligned}
&\omega: \ /(45)\ A \\
&p_1: \ A \quad \rightarrow \ F(l)\ [\&(a)F(l)A]\ /(d)\ [\&(a)F(l)A]\ /(d)\ [\&(a)F(l)A] \\
&p_2: \ F(l) \quad \rightarrow \ F(l * l_r)
\end{aligned}
\tag{2}
$$

Einfluss durch Tropismus

- Tropismus:Tendenz einer Pflanze in eine bestimmte Richtung zu wachsen
- Einfluss wird als Vektor $\overrightarrow{T} \in \mathbb{R}^3$ angegeben
- Beeinflusst die Bewegung der Turtle in Abhängigkeit des Beugungsfaktors $e \in \mathbb{R}$



$$\overrightarrow{T} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, e = 0$$



$$\overrightarrow{T} = \begin{pmatrix} 0 \\ 1 \\ -0.5 \end{pmatrix}, e = 0.27$$

Repräsentation der Turtle-Aktionen als graphentheoretischer Baum

- Turtle-Interpretation baut einen graphentheoretischen Baum $G = \langle V, E \rangle$ auf
- Jeder Knotenpunkt $v \in V$ entspricht einem Punkt $\overrightarrow{p}_v \in \mathbb{R}^3$
- Zustand der Turtle ist ein Tupel $(\overrightarrow{p}, v, \boldsymbol{R})$ mit $v \in V$

- Erweiterung der Turtle-Bewegung $\boldsymbol{F(l)}$ ausgehend von Zustand $(\overrightarrow{\boldsymbol{p}}, \boldsymbol{v}, \boldsymbol{R})$:
  - Bewegung zu Position $\overrightarrow{\boldsymbol{p_{neu}}}$
  - Erstellung eines Knotens $\boldsymbol{v_{neu}}$ und einer Kante $(\boldsymbol{v}, \boldsymbol{v_{neu}})$
  - Neuer Turtle-Zustand: $(\overrightarrow{\boldsymbol{p_{neu}}}, \boldsymbol{v_{neu}}, \boldsymbol{R})$

3 Space Colonization Algorithmus

- Ursprünglich entwickelt zur Darstellung von Blattvenen
- Simuliert Konkurrenzverhalten von wachsenden Zweigen um Wachstumsraum
- Baut graphentheoretischen Baum auf
- Anschließende Nachbearbeitung und Visualisierung der Daten

3.1 Aufbau

- Aufbau

4 Implementierung

- TODO

TODO

- TODO

5 Ergebnisse

5.1 TODO

6 Zusammenfassung und Ausblick

6.1 TODO

Literatur

[Bak] Baker, Martin John: *Maths - Angle between vectors*. http://www.euclideanspace.com/maths/algebra/vectors/angleBetween/index.htm.

[Bal98] Balzert, Helmut: *Lehrbuch der Software-Technik : Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag GmbH, 1998.

[Bec05] Becker, Pete: *Working Draft, Standard for Programming Language C++*, 2005. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1905.pdf.

[Blo85] Bloomenthal, Jules: *Modeling the mighty maple*. Computer Graphics Laboratory, New York Institute of Technology, Old Westbu-

ry, New York, 1985. https://pdfs.semanticscholar.org/00d3/4582edd116a23d4d574ad2c90e9ebf01d74d.pdf.

[DL05] Deussen, Oliver und Bernd Lintermann: *Digital Design of Nature - Computer Generated Plants and Organics*. Springer-Verlag Berlin Heidelberg 2005, 2005.

[EKH10] Eberhardt, Henning, Vesa Klumpp und Uwe D. Hanebeck: *Density Trees for Efficient Nonlinear State Estimation*, 2010. http://isas.uka.de/Publikationen/Fusion10_EberhardtKlumpp.pdf.

[Eng] *Unreal Engine Documentation : Engine Features*. https://docs.unrealengine.com/latest/INT/Engine/index.html.

[FGR] Fink, Prof. Dr. Siegfried, Jörg Grüner und Dr. Christian Rabe: *Skript zum Kernblock „Forstbotanik und Baumphysiologie II “ – Forstbotanischer Teil*. https://www.forstbotanik.uni-freiburg.de/Lehre/Skripten/Skript%20Forstbotanik%20II.

[Gre] *Green One - A landmark render of XfrogPlants by Jan Walter Schliep.* http://xfrog.com/gallery/landscapes/green01_big-1600small.jpg.php.

[GSJ04] Goldman, Ron, Scott Schaefer und Tao Ju: *Turtle Geometry in Computer Graphics and Computer Aided Design*, 2004. http://www.cs.wustl.edu/~taoju/research/TurtlesforCADRevised.pdf.

[LN02] Lefebvre, Sylvain und Fabrice Neyret: *Synthesizing Bark*, 2002. http://www-evasion.imag.fr/Publications/2002/LN02/bark.pdf.

[Lux14] Lux, Prof. Dr. Andreas: *Algorithmen und Datenstrukturen - Vorlesungsskript Kapitel 4*, 2014.

[Man83] Mandelbrot, Benoit B.: *The Fractal Geometry of Nature*. W. H. Freeman and Company, 1983.

[PL90] Prusinkiewicz, Przemyslaw und Aristid Lindenmayer: *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, eBook Auflage, 1990. http://algorithmicbotany.org/papers/abop/abop.pdf.

[Proa] *Procedural Mesh Component in C++ : Getting Started*. https://wiki.unrealengine.com/Procedural_Mesh_Component_in_C%2B%2B:Getting_Started.

[Prob] *Profiling, How To Count CPU Cycles Of Specific Blocks Of Your Game Code*. https://wiki.unrealengine.com/Profiling,_How_To_Count_CPU_Cycles_Of_Specific_Blocks_Of_Your_Game_Code.

[Ran] *Unreal Engine 4 Documentation : Random Streams - Initial Seed*. https://docs.unrealengine.com/latest/INT/Engine/Blueprints/UserGuide/RandomStreams/#initialseed.

[RFL+05] Runions, Adam, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan und Przemyslaw Prusin-

KIEWICZ: *Modeling and visualization of leaf venation patterns*, 2005. http://algorithmicbotany.org/papers/venation.sig2005.pdf.

[RLP07] RUNIONS, ADAM, BRENDAN LANE und PRZEMYSLAW PRUSINKIEWICZ: *Modeling Trees with a Space Colonization Algorithm*, 2007. http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf.

[Sch14] SCHMITZ, PROF. DR. HEINZ: *Theoretische Informatik - Vorlesungsskript*, 2014.

[STN16] SHAKER, NOOR, JULIAN TOGELIUS und MARK J. NELSON: *Procedural Content Generation in Games*. Springer International Publishing Switzerland 2016, 2016.

[Sura] SURIDGE, JAYELINDA: *Modelling by numbers: Part One A: An introduction to procedural geometry*. http://www.gamasutra.com/blogs/JayelindaSuridge/20130903/199457/Modelling_by_numbers_Part_One_A.php.

[Surb] Suridge, Jayelinda: *Modelling by numbers: Part Two A: The cylinder*. http://www.gamasutra.com/blogs/JayelindaSuridge/20130905/199626/Modelling_by_numbers_Part_Two_A.php.

[TKSY] Togelius, Julian, Emil Kastbjerg, David Schedl und Georgios N. Yannakakis: *What is Procedural Content Generation? Mario on the borderline* . http://julian.togelius.com/Togelius2011What.pdf.

[Unra] *Unreal Engine Documentation : Content Examples*. https://docs.unrealengine.com/latest/INT/Resources/ContentExamples/.

[Unrb] *Unreal Engine Documentation : Unreal Engine 4 Terminology*. https://docs.unrealengine.com/latest/INT/GettingStarted/Terminology/index.html.

[Wha] *Unreal Engine Features*. https://www.unrealengine.com/unreal-engine-4.